Locally Non-rigid Registration for Mobile HDR Photography

Orazio Gallo[1] Alejandro Troccoli[1] Jun Hu[1,2] Kari Pulli[1,3] Jan Kautz[1]
[1]NVIDIA [2]Duke University [3]Light

(a) Our HDR result (b) Single homography (c) Bao et al. (d) Our result

Figure 1: When capturing HDR stacks without a tripod, parallax and non-rigid scene changes are the main sources of artifacts. The picture in (a) is an HDR image generated by our algorithm from a stack of two pictures taken with a hand-held camera (notice that the volleyball is hand-held as well). A common and efficient method to register the images is to use a single homography, but parallax will still cause ghosting artifacts, see (b). One can then resort to non-rigid registration methods; here we use the fastest method of which we are aware, but artifacts due to erroneous registration are still visible (c). Our method is several times faster and, for scenes with parallax and small non-rigid displacements, produces better results (d).

Abstract

Image registration for stack-based HDR photography is challenging. If not properly accounted for, camera motion and scene changes result in artifacts in the composite image. Unfortunately, existing methods to address this problem are either accurate, but too slow for mobile devices, or fast, but prone to failing. We propose a method that fills this void: our approach is extremely fast—under 700*ms on a commercial tablet for a pair of 5MP images—and prevents the artifacts that arise from insufficient registration quality.*

1. Introduction

High-Dynamic-Range (HDR) imaging has become an essential feature for camera phones and point-and-shoot cameras—even some DSLR cameras now offer it as a shooting mode. To date, the most popular strategy for capturing HDR images is to take multiple pictures of the same scene with different exposure times, which is usually referred to as *stack-based* HDR. Over the last decade, the research community also proposed several hardware solutions to sidestep the need for multiple images, but the trade-off between cost and picture quality still makes stack-based strategies more appealing to camera manufacturers.

Combining multiple low-dynamic-range (LDR) images into a single HDR irradiance map is relatively straightforward, provided that each pixel samples the exact same irradiance in each picture of the stack. In practice, however, any viable strategy to merge LDR images needs to cope with both camera motion and scene changes. Indeed there is a rich literature on the subject, with different methods offering a different compromise between computational complexity and reconstruction accuracy.

On one end of the spectrum there are light-weight methods, generally well-suited to run on mobile devices. These methods address the problem of camera motion by estimating a global transformation in a robust fashion [19, 17]. After image alignment, scene changes can be addressed with some flavor of outlier rejection, often called deghosting. This can be achieved by picking one image of the stack to act as a reference and only merging consistent pixels from the other images [4, 15]. Alternatively, for sufficiently large stacks, one can merge only the irradiance values most often seen for a given pixel [20, 14]. The price for the computational efficiency of rigid-registration methods is their severe limitation in terms of accuracy: even the most general global transformation, *i.e.*, a homography, cannot correct for parallax, which occurs for non-planar scenes every time

the camera undergoes even a small amount of translation.

On the other end of the spectrum lie methods that allow for a completely non-rigid transformation between the images in the stack [16, 9]. Rather than separating camera motion and scene changes, these algorithms attempt to "move" any given pixel in one shot of the stack to its corresponding location in the reference image. These methods have shown impressive results, essentially with any amount and type of motion in the scene, but generally require minutes on desktop computers: they are simply impractical for deployment on mobile devices.

We fill the gap between these two extreme points in the space of registration accuracy versus computational complexity. Our work builds on the observation that most modern devices, such as the NVIDIA SHIELD Tablet or the Google Nexus 6 phone, are capable of streaming full-resolution YUV frames at 30fps. Given an exposure time t, the delay between consecutive shots is then $(33 - t)$ms $<$ 33ms[1], which prevents large displacements of moving objects. Parallax, however, remains an issue even for small camera translations. Figure 1(b) shows the extent of these artifacts. (Note that, for better visualization, all the insets in Figure 1 were generated with a simple blending.)

Our method comprises a strategy to find sparse correspondences that is particularly well-suited for HDR stacks, where large parts of some images are extremely dark. After detecting spurious matches, it corrects for parallax by propagating the displacement computed at the discrete locations in an edge-aware fashion. The proposed algorithm can also correct for small non-rigid motions, but may fail for cases where the subject simply cannot be expected to cooperate, as is the case in sport photography. To address such cases, we couple our locally non-rigid registration algorithm with a modified version of the exposure fusion algorithm [13].

Our method runs in 677ms on a stack of two 5MP images *on a mobile device*, which is several orders of magnitude faster than any non-rigid registration method of which we are aware. On a desktop machine, our method can register the same stack in 150ms, corresponding to a speedup of roughly $11\times$ over the fastest optical flow methods published to date (see Section 3).

2. Method

Several recently published methods successfully tackled the task of non-rigid registration for large displacements by using approximate nearest neighbor fields (NNFs) [9, 16]. While the quality of the results they produce is impressive, even in the case of very large displacements, their computational cost is prohibitive, in particular for mobile devices. Moreover, given the frame rate at which bursts of images can be acquired, the tolerance to large displacements that those methods offer is most often unnecessary.

The problem of large displacements is further attenuated by the dynamic range of modern sensors, which allows to capture most scenes with only two shots; leveraging on this observation, we focus on two-image exposure stacks, although the extension to more images only requires to run the algorithm $n - 1$ times for a stack of n images, as shown in Figure 5, where we register a stack of three images. Rather than computing an expensive NNF, which, for the vast majority of stacks, would mostly consist of small and relatively uniform displacements, we find sparse correspondences between the two images. While extremely fast, the matcher we designed for this purpose produces accurate matches, even in extremely dark regions—a particularly important feature for HDR stacks. To solve the parallax problem, rather than registering the images with a single homography, we propose to propagate the sparse flow from the matches computed in the previous stage in an edge-aware fashion. To merge the images we modified exposure fusion [13] to compensate for potential errors in the computation of the flow. We implemented the full pipeline—stack capture, image registration, and image fusion—on an NVIDIA SHIELD Tablet.

In the remainder of this section we describe in detail the different components of our algorithm.

2.1. Stack capture and reference selection

Metering for HDR, *i.e.*, the selection of the exposure times and number of pictures required to sample the irradiance distribution for a particular scene, has been an active area of research [5, 7, 8]. We observe that the dynamic range of modern sensors allows to capture most real-world scenes with as little as two exposures, and devise a simple strategy that works well in our experiments: we use the Expose-To-The-Right (ETTR) paradigm [8] for the first image in the stack, and select the second exposure time to be 2, 3, or 4 stops brighter, based on the number of under-exposed pixels (the more under-exposed pixels, the longer the second exposure). Limiting the number of candidate exposures to three allows for a faster metering; moreover, the advantage of a higher granularity is difficult to appreciate by visually inspecting the HDR result.

Then, rather than picking the reference image for our registration algorithm to be the one with the least saturated and underexposed pixels [4], we always use the shortest (darkest) exposure as the reference; this is because, while the noise in the dark regions of a scene makes it difficult to find reliable matches, saturation makes it impossible. In the rest of the paper we will refer to the two images in the stack as *reference* and *source*, indicating our final goal to warp the source to the reference.

[1] If $t > 33$ms, the limiting factor will likely be blur, and the delay between shots will still be $(33 - \mathrm{mod}(33, t))$ms < 33ms.

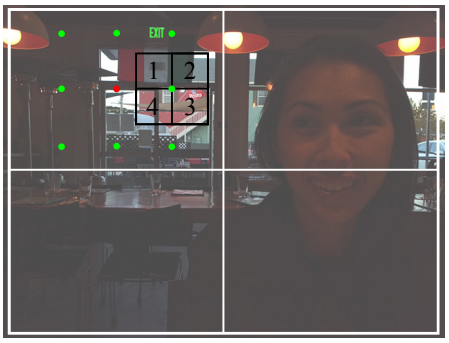


Figure 2: After dividing the reference image in tiles, our matcher looks at the center of each tile (red dot) and a set of predefined locations around it (green dots). It then computes a measure of *cornerness* based on the average luminance in the four quadrants around each candidate corner.

2.2. A fast, robust matcher

In order to produce fast and reliable correspondences, even in the presence of the noise the potentially short exposure time induces, we propose a novel matcher. To this end, we efficiently find distinct features, *i.e.*, corners, in the reference image, which we then match in the second image with a patch-based search. First, we define a simple measure of *cornerness*:

$$C(p) = \sum_{j=1}^{4} |\mu_{\text{mod}(j+1,4)} - \mu_j| \tag{1}$$

where p is the pixel location, and μ_j is the average luminance value in the j^{th} quadrant around p, marked in black in Figure 2. Equation 1 simply measures the change in the average luminance in the four quadrants around p. However, for p to be a good corner candidate, we also require that the minimum difference of average luminance between any two contiguous quadrants be large:

$$\min_{j \in \{1,\dots,4\}} |\mu_{\text{mod}(j+1,4)} - \mu_j| > T \tag{2}$$

where T is a given threshold. Essentially, Equation 2 prevents situations in which a point on an edge is promoted to a corner. Note that, regardless of the number of pixels in each quadrants, μ_j can be computed very efficiently using integral images.

To encourage a uniform distribution of corners over the image, we first divide the reference image in tiles. Then, we look at a predefined set of locations (and not at all pixels for computational efficiency) around the center of each tile (Figure 2), and retain the one with the highest cornerness value. Note that Equation 2 also prevents finding corners in flat regions, therefore tiles that are completely flat may not be assigned a corner.

Although the proposed algorithm to search for corners is extremely efficient, we do not run it on both reference and source images, as it only serves as a feature detector. Hence, we evaluate corners in the reference image only, and then search for matches within a predefined search radius around the corresponding position in the source image using the sum of squared differences metric (SSD).

Finally, to minimize the computational cost while allowing for a large search radius, we use a pyramidal approach. At level ℓ of the pyramid, we find a set of corners $\boldsymbol{x}^{\ell}_{\text{ref}}$ in the reference. For each corner $\boldsymbol{x}^{\ell}_{\text{ref}}$ we search for matches in the source image within a search radius around

$$\boldsymbol{x}^{\ell}_{\text{src}} = H^{\ell+1} \boldsymbol{x}^{\ell}_{\text{ref}}, \tag{3}$$

where $H^{\ell+1}$ is a single homography computed at the previous layer of the pyramid using its corners and matches. Equation 3 holds if both $\boldsymbol{x}_{\text{src}}$ and $\boldsymbol{x}_{\text{ref}}$ are represented in a normalized coordinate system such that $x \in [-1, 1]$ and $y \in [-h/w, h/w]$, where $\boldsymbol{x} = (x, y)$, and w and h are width and height of the image. We also move the origin to the center of the image. This homography only serves as a way to initialize the search locations in the source image (and in turn reduces the required search radius compared to a random initalization). Note that $\boldsymbol{x}^{\ell}_{\text{ref}}$ are computed directly on layer ℓ, and not upsampled from layer $\ell + 1$, as relevant features at layer ℓ may have not been present at layer $\ell + 1$ and may not be visible in layer $\ell - 1$.

2.3. Weeding out spurious matches

The matcher we describe in Section 2.2 generally produces robust matches even in very low light conditions (see Section 3 for a more detailed evaluation). However, to detect and remove potential spurious matches, we run an additional filtering stage. The key idea is that we require matches to be locally consistent with a homography; those that are not, are likely to be incorrect, because we expect small displacements. The consistent matches can be determined by means of a robust method, such as RANSAC [3]. A straightforward application of RANSAC, however, is too expensive. Instead, we developed a fast filtering strategy to weed out spurious matches that are not consistent with a homography.

The goal of our novel filter is to efficiently obtain the set M of reliable matches. We consider a match to be reliable if there exists a large number of matches that are mapped from the source image to the reference image by the same local homography H; in other words, we aim at finding all the matches that induce any homography supported by a large set of inliers.

The filter, inspired by RANSAC, works iteratively. Specifically, at the i^{th} iteration, we randomly sample 4 points from our set of matches, fit a homography H^i, and find the subset of inliers I^i with respect to H^i. Then, rather

than saving the homography supported by the largest set of inliers, we simply update M using the following rule:

$$M^i = \begin{cases} M^{i-1} \cup I^i & \text{if } |I^i| > \delta \\ M^{i-1} & \text{otherwise} \end{cases}, \quad (4)$$

where $|\cdot|$ indicates the cardinality of a set, and δ is a threshold. To understand the idea behind Equation 4, consider a toy scene where most of the corners are distributed on two static planar surfaces at different distances from the camera, with a few other moving non rigidly. We would like our algorithm to weed out the non-rigid corners, as well as the corners on the two surfaces that are incorrectly matched. At each iteration, one of two things can happen. First, the sampling may include corners from both planes or those moving non-rigidly; the resulting homography will have a small number of inliers, and the set of reliable matches M will not be modified. Second, all of the points sampled at the current iteration belong to one of the two planes. In this case the resulting homography will explain the motion of all the corners that are on the same plane and that move rigidly; the set of reliable matches M will be updated to include these inliers. Note that we do not remove the inliers of the i^{th} iteration from the original set of corners.

We further speed up the process by running n instances of our filter2 on separate threads, with each instance running only $1/n$ iterations; because we take the union of the acceptable inliers from previous iterations, running our filter n times for $1/n$ iterations is exactly equivalent to a single run on n iterations. After each run k terminates, we simply merge the resulting sets M_k.

2.4. Sparse-to-dense flow

So far, we have described a method to efficiently find a set of robust matches between the images, constituting sparse flow. To be able to warp the source image to the reference, however, we need to compute the displacement at every pixel. A simple interpolation of the sparse flow would produce artifacts similar to those caused by using a single homography: depth discontinuities and boundaries of moving objects would not be aligned accurately.

Instead, we would like to interpolate the sparse flow in an edge-aware fashion. The problem is similar to that of image colorization [10], where colors are propagated from a handful of sparse pixels that have been assigned a color manually. In our case, we propagate the flow components (u, v) computed at discrete locations.

For this purpose, we employ an efficient CUDA implementation of the algorithm proposed by Gastal and Oliveira [6], and use it to cross-bilateral filter the flow. Similarly to how they propose to propagate colors, we first create two maps P_u and P_v

$$P_f(p) = \begin{cases} f(p) & \text{if } p \text{ is a corner} \\ 0 & \text{elsewhere} \end{cases}, \quad (5)$$

where $f = \{u, v\}$. We then use the reference image to cross-bilateral filter the maps. However, while this propagates the flow in an edge-aware fashion generating the two maps $\widetilde{P}_f$, it will affect the value of the flow at the location of the corners, which should not change. Therefore, we use a normalization map

$$N(p) = \begin{cases} 1 & \text{if } p \text{ is a corner} \\ 0 & \text{elsewhere} \end{cases}. \quad (6)$$

The final flow F can then be computed as $F_f = \widetilde{P}_f / \widetilde{N}$, where $\widetilde{N}$ is the cross-bilateral filtered version of N.

2.5. Error-tolerant image fusion

If the number of matches between reference and source images is low, or if a particular area of the scene is textureless, the quality of the flow propagation described in Section 2.4 can deteriorate because the accuracy of the flow is affected by the spatial distance over which it needs to propagate. To detect and compensate for errors that may arise in such cases, we propose a simple modification of the exposure fusion algorithm proposed by Mertens *et al.* [13]. In addition to weights for contrast, color saturation, and well-exposedness, we add a fourth weight that reflects the quality of the registration. Specifically, we choose to use the structural similarity index (SSIM) [18]. Note that computing the SSIM map only requires to perform five convolutions with Gaussian kernels and a few other parallelizable operations such as pixel-wise image multiplication and sum; this makes a GPU implementation of SSIM extremely efficient, see Section 3 for an analysis of its runtime.

Figure 3 shows an example of failure of the edge-aware propagation stage, and how our error-tolerant fusion can detect and compensate for it.

2.6. Implementation details

For the matcher, we create up to 5 pyramid layers (we stop early if the coarsest level falls below 100 pixels in either height or width). The patch comparison is computed on 21×21 patches, and the maximum search radius at each level is 10. We evaluate the cornerness within a patch on a regular grid of points spaced by $1/16^{\text{th}}$ of the tile size. We implemented the method described here—from capture to generation of the HDR image, with a mixture of C++ and CUDA code. Specifically, the matcher and the weeding stage run on the CPU, with the weeding stage being multi-threaded. The remaining parts are heavily CUDA-based. Finally, for the sparse-to-dense stage, it is important to use

(a) Blended stack (b) Source warped (c) Final HDR

Figure 3: A failure case of the flow propagation stage. The input images are first blended to show the original displacement (a). The warped source produced by our algorithm presents a few artifacts, a couple of which are marked. However, our error-tolerant exposure fusion can detect and correct for those errors.

a large spatial standard deviation to make sure that the flow can be propagated to regions poor in number of correspondences; we use $\sigma_s = 400$.

3. Evaluation and Results

In this section we evaluate the performance of our algorithm, both in terms of quality of the result and execution time, by means of comparisons with state-of-the-art methods. Note that we perform histogram equalization on all the input images to attenuate brightness differences.

3.1. Quality comparisons

We are particularly interested in evaluating the quality of our matcher, and the quality of the final result when compared against other non-rigid registration methods.

The matcher—One of our claims pertains to the robustness of our matcher in particularly low-light situations. Figure 4 shows a comparison between our matcher and SIFT [11]. In terms of robustness, SIFT is arguably the state-of-the-art method for finding correspondences between images. And indeed it can produce reliable correspondences even in the presence of large displacements, where our matcher would fail. However, when one of the two images is extremely dark, SIFT may fail dramatically, as shown in Figures 4c and 4h. One can filter them in a manner similar to the one we propose in Section 2.3. Nevertheless, in extreme cases such as those shown in the figure, the correspondences may be so poor that after the filtering stage, too few are left to perform an accurate warp; the low number and quality of the correspondences shown in Figures 4d and 4i for instance, are the cause of the artifacts visible in the first and second row of Figure 8d. On the contrary, our method still produces high-quality correspondences. This ability is key to the success of the registration of HDR stacks.

Non-rigid registration algorithms—The context of our method is different from that of algorithms that aim at achieving a high-quality result, even in the presence of large displacements. However, we still compare on cases that are within the scope of our paper; for the comparison we pick the algorithms that can deliver the best quality [16, 9], and the fastest non-rigid registration algorithm of which we are aware [1].

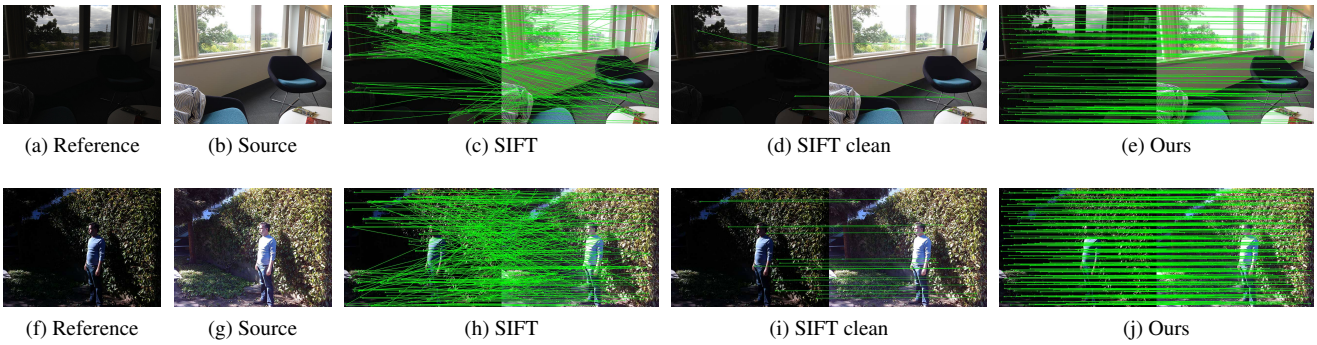
(a) Reference (d) Sen *et al*. [16]

(b) Source 1 (e) Hu *et al*. [16]

(c) Source 2 (f) Our result

Figure 5: Comparison with the state-of-the-art methods for non-rigid registration. To produce a result that is visually comparable to the related work, we use the tonemapping operator proposed by Mantiuk *et al*. [12], rather than our modified exposure fusion, see Section 3.1; however, the color differences that are still visible are solely due to the tonemapper parameters.

Figure 5 shows a comparison with competitors that deliver the highest quality. To perform it, we registered the images in the stack to the shortest exposure. Note that "Source 2" is 4 stops brighter than the reference, and yet our method correctly warps it; the other two methods use the middle exposure as the reference. Also, to simplify the task of visually comparing the results of the three approaches, rather than using the modified version of exposure fusion that we described in Section 2.5, we output the warped images, create an HDR irradiance map, and use the tonemapper proposed by Mantiuk *et al*. [12]. The quality of the sky in our result is comparable with that of Hu *et al*., and better than that of Sen *et al*.—the sun is still present and there are no halos. Note that some of the people walking under the dome are not correctly registered by our method; both the other results correctly register that region. However, as mentioned



(a) Reference (b) Source (c) SIFT (d) SIFT clean (e) Ours

(f) Reference (g) Source (h) SIFT (i) SIFT clean (j) Ours

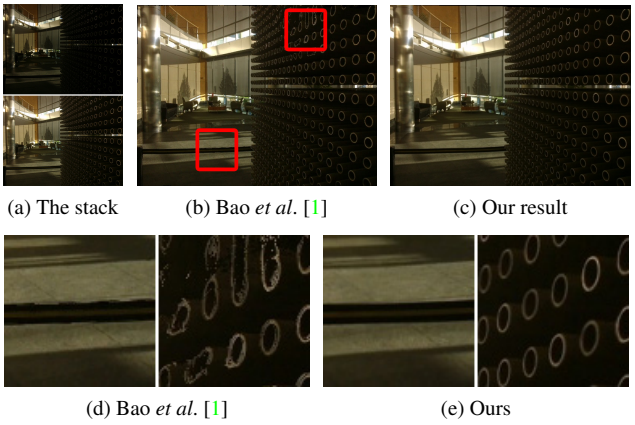
Figure 4: The matcher we propose performs particularly well when searching for correspondences in extremely dark areas, as is needed for large portions of the two stacks shown here. SIFT fails to find reliable correspondences; a solution could be to only retain the matches that support a homography, here indicated as "SIFT clean". However, if the quality of the original matches is too low, very few correspondences survive the cleaning stage, as is the case shown in (d) and (i). Our method produces a uniformly distributed set of matches. Note that both methods were fed images that were histogram equalized.

above, in this example we did not run our error-tolerant fusion, which would take care of that problem.

A method more similar in spirit to ours, is the flow algorithm recently proposed by Bao *et al*. [1]. While not specifically designed for HDR registration, their algorithm is impressively fast (see Section 3.2). At its core, the method by Bao and colleagues uses PatchMatch to deal with large displacements [2]. To ameliorate the flow accuracy in occlusion and disocclusion regions, they compute the matching cost in an edge-aware fashion; at the same time they improve on speed by computing the cost only at a wisely selected subset of pixels. Note that, despite being several times faster than the competitors, the method by Bao and colleagues ranks within the top ten positions in all of the established flow benchmark datasets. We compare our method with theirs on cases that are within the scope of both algorithms.

Figure 1 shows a fairly common case for an HDR stack, with both camera motion and slight scene motion (the woman is holding the volleyball). In all the comparisons with their method, we first equalize the images to compensate for illumination changes. The method by Bao and colleagues produces strong artifacts that are visible in Figure 1(c); on the contrary our method registers the images perfectly. Note that the original images are 5MP images, which is possibly larger than what their method was originally designed for; please see the additional material for more comparisons, including lower resolution stacks.

Figure 6 shows another comparison, this time with both algorithms running on a VGA stack. In order to perform a more fair comparison, the images were taken with a small, 1-stop separation, and neither of them presents saturation; because of the limited dynamic range and spacing of the exposure times of this example, histogram equalization makes



(a) The stack (b) Bao *et al*. [1] (c) Our result

(d) Bao *et al*. [1] (e) Ours

Figure 6: Comparison with the method by Bao *et al*. [1]. Images (b) and (c) are the source images warped with the method by Bao *et al*. and by our algorithm respectively. Notice the artifacts affecting the results by Bao *et al*.

the source and the reference essentially identical in terms of brightness. The insets of the figure show that the method by Bao *et al*. fails in preserving the local structure of the tubes.

On both examples, our algorithm produces a more accurate registration. Figure 8 shows more results of our method.

3.2. Execution time

One of the biggest strengths of our method is its computational efficiency. We first validate this claim by comparing the runtime of our algorithm to three related works. For this experiment, we used VGA images. Two preliminary comments are in order; first, the methods by Sen *et al*. and Hu *et al*. are implemented in a mixture of Matlab and C++

Algorithm	Execution time	Speedup
Our algorithm	**49ms**	—
Bao *et al*. [1]	171ms	$\approx 3.5\times$
Sen *et al*. [16]	106^*s	$> 1,900^*\times$
Hu *et al*. [9]	94^*s	$> 2,000^*\times$

Table 1: Comparison of the execution time with different state-of-the-art algorithms. The tests were run on **VGA** images. The * indicates execution times for a mixture of Matlab and C++ code.

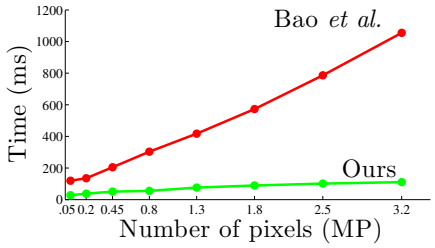


Figure 7: Computational time of the algorithm by Bao *et al*. [1] and ours. Note that our method grows sublinearly. The timings were captured on an NVIDIA GTX Titan graphics card.

code, which makes them intrinsically slower. However, the speedup is significant even when accounting for that. Second, the execution times shown in Table 1 for their methods are those reported by Oh *et al*. [14].

A more interesting comparison is with the method of Bao and colleagues, both because they implemented their algorithm very efficiently in CUDA, and because execution speed is one of their main focuses. Indeed, recall that, to the best of our knowledge, theirs is the fastest published method for optical flow. And yet, our method is roughly $3.5\times$ faster. This, however, is only a partial evaluation: while the execution time of our algorithm grows sublinearly, theirs grows linearly with the number of pixels, as shown in Figure 6. On an NVIDIA GTX Titan, for a pair of 5MP images, their code runs in 1.66s; our method registers the same images in 150ms, which translates to a speedup $\mathbf{11}\times$.

Table 2 shows the cost of each step our algorithm on a desktop machine as well as a tablet, both for pairs of 5MP images. Aside from rigid registration methods, we are not aware of any published work capable of registering two 5MP images in a time even close to a second on a desktop. Our approach can do it in less than that (677ms) on a commercial tablet.

Moreover, as shown in Figure 7, our method scales well with image size; this is a particularly attractive feature, given the rate at which the number of pixels in widely available sensors is growing.

Step of the algorithm	Tablet	Desktop
Matcher (Sec. 2.2)	132ms	49ms
Match weeding (Sec. 2.3)	23ms	20ms
Sparse-to-dense flow (Sec. 2.4)	473ms	67ms
Fusion weights (Sec. 2.5)	49ms	11ms
Total time	**677ms**	**147ms**

Table 2: Computational time for each step of the algorithm when run on a pair of **5MP** images. The reference tablet is an NVIDIA Shield Tablet, which is equipped with a Tegra K1 system-on-chip. The timings on desktop were measured on an Intel I7 CPU with an NVIDIA GTX Titan graphics card.

4. Conclusions

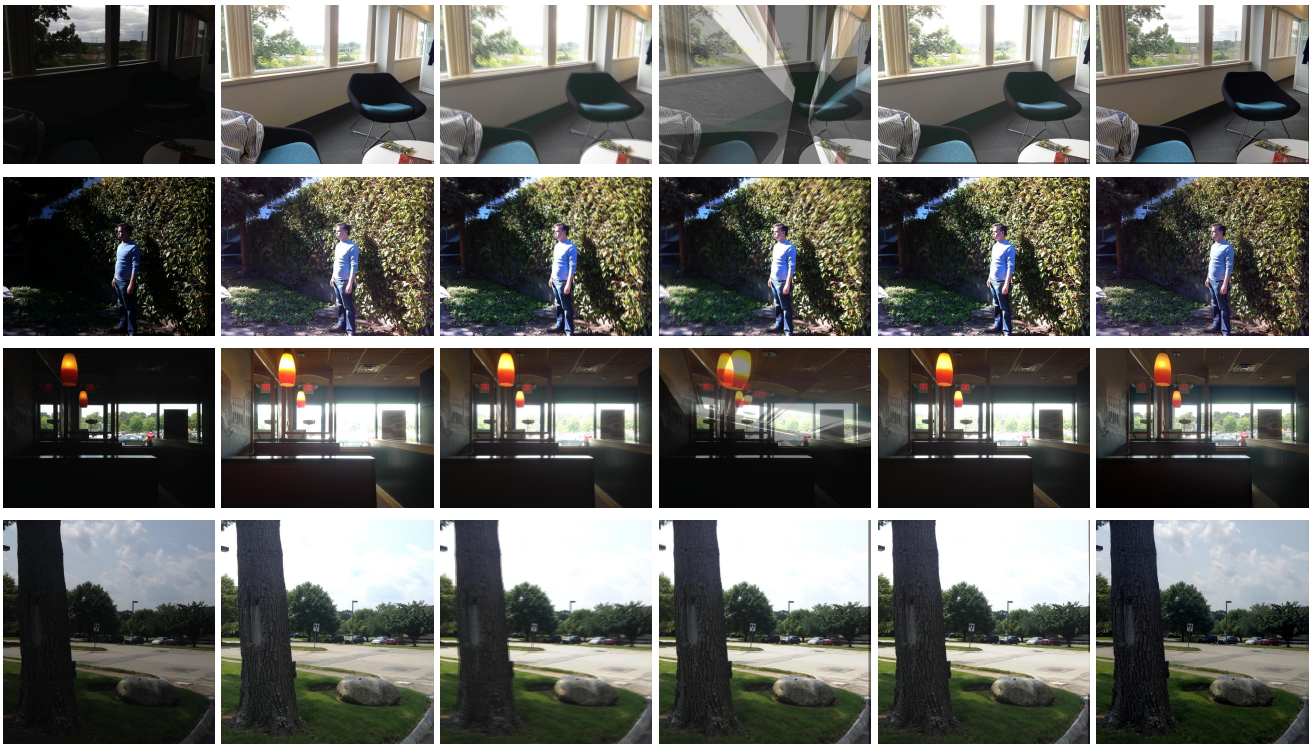
In the space of registration for HDR imaging, and stack-based photography in general, it is difficult to find an acceptable trade-off between registration accuracy and computational load. We propose a new compromise: rather than attempting to solve the most general non-rigid registration case, we focus on the more typical case of relatively small displacement, and propose a locally non-rigid registration technique. Specifically, we contribute a method that is $11\times$ faster than the fastest published method, while producing a more accurate registration. Our approach is also the only one that can perform non-rigid registration within the computational power of a mobile device. To achieve this result, we developed a novel, fast feature matcher that works better than the state-of-the-art when the reference image is underexposed. Our matcher comprises an original light-weight corner detector, and a matching strategy based on a modification of the RANSAC algorithm. We think that this matcher may be useful for other applications as well. Finally, we implement the complete system, from capture to HDR generation, on an NVIDIA SHIELD Tablet. This also involves a metering strategy, a flow propagation step, and a deghosting strategy to compensate for errors in the flow propagation. vspace-1mm

Acknowledgments

The authors would like to thank Colin Tracey and Huairuo Tang for their help in writing part of the early CUDA implementation.

References

[1] L. Bao, Q. Yang, and H. Jin. Fast edge-preserving PatchMatch for large displacement optical flow. In *CVPR*, 2014. 5, 6, 7



(a) Reference (b) Source (c) Blended Stack (d) Homogr. + Blend (e) Our Align. + Blend (f) Our Final HDR

Figure 8: Our method works well on a variety of scenes. In (a) and (b) we show the two input images. Column (c) shows the stack directly blended together, without further alignment. In (d), we matched SIFT features and computed a single homography with RANSAC to align the images; to better show the displacements they are simply blended. Column (e) shows our alignment of the two images, again visualized using a simple blend. Finally, column (f) shows the resulting HDR image.

[2] C. Barnes, E. Shechtman, D. B. Goldman, and A. Finkelstein. The generalized PatchMatch correspondence algorithm. In *ECCV*, 2010. 6

[3] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 1981. 3

[4] O. Gallo, N. Gelfand, W. Chen, M. Tico, and K. Pulli. Artifact-free high dynamic range imaging. In *ICCP*, 2009. 1, 2

[5] O. Gallo, M. Tico, R. Manduchi, N. Gelfand, and K. Pulli. Metering for exposure stacks. *Eurographics*, 31:479–488, 2012. 2

[6] E. S. Gastal and M. M. Oliveira. Domain transform for edge-aware image and video processing. In *ACM Transactions On Graphics*, volume 30, 2011. 4

[7] M. Granados, B. Ajdin, M. Wand, C. Theobalt, H.-P. Seidel, and H. Lensch. Optimal HDR reconstruction with linear digital cameras. In *CVPR*, 2010. 2

[8] S. W. Hasinoff, F. Durand, and W. T. Freeman. Noise-optimal capture for high dynamic range photography. In *CVPR*, 2010. 2

[9] J. Hu, O. Gallo, K. Pulli, and X. Sun. HDR deghosting: How to deal with saturation? In *CVPR*, 2013. 2, 5, 7

[10] A. Levin, D. Lischinski, and Y. Weiss. Colorization using optimization. In *ACM Transactions On Graphics*, 2004. 4

[11] D. G. Lowe. Object recognition from local scale-invariant features. In *ICCV*, 1999. 5

[12] R. Mantiuk, K. Myszkowski, and H.-P. Seidel. A perceptual framework for contrast processing of high dynamic range images. *ACM Transactions on Applied Perception*, 2006. 5

[13] T. Mertens, J. Kautz, and F. V. Reeth. Exposure fusion. In *Pacific Graphics*, 2007. 2, 4

[14] T. Oh, J. Lee, Y. Tai, and I. Kweon. Robust high dynamic range imaging by rank minimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2014. 1, 7

[15] S. Raman and S. Chaudhuri. Reconstruction of high contrast images for dynamic scenes. *The Visual Computer*, 2011. 1

[16] P. Sen, N. K. Kalantari, M. Yaesoubi, S. Darabi, D. B. Goldman, and E. Shechtman. Robust patch-based HDR reconstruction of dynamic scenes. In *SIGGRAPH Asia*, 2012. 2, 5, 7

[17] G. Tzimiropoulos, V. Argyriou, S. Zafeiriou, and T. Stathaki. Robust FFT-based scale-invariant image registration with

image gradients. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2010. 1

[18] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 2004. 4

[19] G. Ward. Fast, robust image registration for compositing high-dynamic-range photographs from handheld exposures. *Journal of Graphics Tools*, 2003. 1

[20] W. Zhang and W.-K. Cham. Reference-guided exposure fusion in dynamic scenes. *Journal of Visual Communication and Image Representation*, 2012. 1